Bolt Beranek and Newman Inc.

Report No. 3752

Distributed Computation and TENEX-Related Activities

Quarterly Progress Report No. 11, 1 May 1977 to 31 July 1977

January 1978

Prepared for:
Defense Advanced Research Projects Agency

BBN Report No. 3752

(14) BBN-3752

(6) DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES.

(10) R. Sahantz Robert H. Thomas

(9) Quarterly Progress Report No. 11,
1 May [redacted] 31 Jul[redacted]77,

(11) Jan[redacted]78 (12) 42P.

(15) NØØØ14-75-C-Ø773, ARPA Order-2935

Sponsored by:

Defense Advanced Research Projects Agency
ARPA Order No. 2935

Monitored by:

Office of Naval Research
Under Contract No. NØØØ14-75-C-Ø773
Contract Period 1 November 1974 to 1 May 1978

Principal Investigator: Robert H. Thomas

060 100 JOB

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 3752	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES		5. TYPE OF REPORT & PERIOD COVERED 5/1/77 - 7/31/77 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. Schantz, R. Thomas		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0773
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, Massachusetts 02138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE January 1978 13. NUMBER OF PAGES 38
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2935.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

distribution computation
National Software Works
distributed operating system
TENEX operating system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes BBN efforts in the design of the National Software Works system and ~~BBN~~ efforts to integrate TENEX into the National Software Works system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Table of Contents

1. Introduction 1

2. MSG: The NSW Interprocess Communication Facility 4

3. The TENEX Foreman 13

Appendix A - Design for the Management of Saved Tool Sessions

1. Introduction

Our participation in the National Software Works (NSW) project continued through this quarter. As noted in our previous quarterly reports, the basic NSW concept has been adequately demonstrated by various prototype versions of the system and the focus of the project has turned from concept demonstration to development of a functional, operational NSW system. Development of an acceptable operational system from these early system versions requires: increased functionality; improved performance and responsiveness; increased reliability; and improved operator tools for system control.

Development of an operational NSW system is a decidely non-trivial task and the effort required to achieve it should not be underestimated. To the best of our knowledge the only similar distributed system to achieve a nearly operational status is the RSEXEC system, and its goals were considerably less ambitious than those of the NSW.

Our efforts this quarter have been directed primarily toward implementation aspects of the project. In particular, we have worked on the TENEX (and TOPS-20) implementation of MSG (the NSW interprocess communication facility), the TENEX Foreman (the tool bearing host software module responsible for controlling NSW tool execution), and the NSW dispatcher (the system component

responsible for creating instances of NSW Front End processes when users initially attempt to access NSW). We expect that our efforts for the next several quarters will continue to focus on implementation issues as we work to develop an operational NSW system.

Work was started this quarter to modify the TENEX MSG implementation so that it will also run under the DEC TOPS-20 operating system. This is one of the steps required to make the DEC System20 an NSW tool bearing host and we expect to complete the MSG conversion during the next quarter. In addition, we have made a number of improvements to the TENEX MSG implementation. The work on MSG is described in detail in Section 2 of this report.

Efforts on the TENEX Foreman component and related tool bearing host software occurred primarily in the following areas: implementation of software to analyze system performance data collected by the Foreman during tool sessions; design and implementation of a "test encapsulator" which will be used as an aid to facilitate the installation of new tools into the NSW system; installation of a variety of new tools; and, improvements to the Foreman component. These and other activities related to the Foreman are discussed in detail in Section 3.

In addition to the implementation work, we attended a number of meetings as part of our participation in the NSW project. In

June we met with personnel from Rome Air Development Center to discuss future directions for the NSW project. We attended a meeting hosted by the Rand Corporation in July on the interprocess communication requirements of the joint ARPA/Navy Advanced Command Control Architectural Testbed (ACCAT) program at which we discussed the interprocess communication problems posed by NSW and described the NSW approach to these problems. At the meeting MSG was adopted as the ACCAT standard for interprocess communication. In June we gave an invited talk on the NSW system, its goals and approaches, at a meeting held in Newport, Rhode Island for the New England chapter of the Federal Information Processing Society. Finally, we continued to interact with DEC to work out the details of the enhancements planned for the DEC TOPS-20 operating system that are required to enable its integration into NSW, both as a tool bearing host and as a Works Manager host.

2. MSG: The NSW Interprocess Communication Facility

During this quarter we started to modify the TENEX MSG implementation so that it will run under the new DEC TOPS-20 operating system (1). This is one of a series of steps required to integrate TOPS-20 into NSW. The basic strategy for accomplishing this integration is to modify the software that was developed to make TENEX an NSW host so that it can run under the TOPS-20 system. This is a feasible approach because the two systems are so similar, both in terms of hardware and operating system software.

The steps required to integrate TOPS-20 into NSW are:

- Enhance the TOPS-20 operating system so that it is capable of supporting NSW software.
  Although TOPS-20 is a direct descendent of a version of TENEX, new features continued to be added to TENEX after the TOPS-20 implementation project began. The TENEX NSW software uses some of these TENEX features which are not currently supported by TOPS-20. These features include

---

1. For a number of years the DEC KA10 processor operating under the TENEX operating system developed at BBN has been central to many ARPA information processing research projects. Recently DEC has developed a new line of processors, including the KL10 and KL20 processors, which are significantly faster than, but functionally compatible with, the older KA10. The operating system for this new line of processors is a descendent of an early version of TENEX (circa 1972) and is called TOPS-20. The new DEC processor together with the TOPS-20 operating system is called the DEC System20. Because the DEC System20 offers similar capabilities to the older KA10-based TENEX system and is considerably more cost effective, many ARPA sponsored installations, including some that will support NSW tool bearing hosts, have been upgrading their equipment to these newer systems.

JSYS traps, fork groups, and various options of the CRJOB JSYS. The work to enhance TOPS-20 will be performed by BBN under a separate contract.

- Modify TENEX MSG to run under the TOPS-20 system.
  The TOPS-20 implementation of MSG will provide the communication services needed by the other NSW components (e.g., the Foreman) which are required on TOPS-20 to make it a tool bearing host.

- Modify the TENEX Foreman and related software to run under TOPS-20.
  This will allow TOPS-20 software packages to be run as NSW tools under the control of a TOPS-20 Foreman, which itself accepts direction from the NSW Works Manager.

- Modify the TENEX File Package to run under TOPS-20.
  A File Package for TOPS-20 is necessary to permit TOPS-20 tools to access and create NSW files.

- Modify, as appropriate, other NSW software written for TENEX.
  Because DEC System20 is more cost effective than TENEX, it makes sense to use TOPS-20 to support other NSW system components which are currently supported by TENEX. For example, it is likely that NSW performance could be significantly improved by running the Works Manager on a DEC System20 host rather than TENEX.

The reason it is necessary to modify the various TENEX NSW components is that the two systems, while very similar, are not identical. The enhancements to TOPS-20 noted above will make the two systems more or less functionally equivalent in the areas that are important to NSW. However even after these enhancements have been made, there will still be differences in the program interfaces to operating system functions (JSYSs) and in the detailed operation of these functions.

Our goal in converting the TENEX NSW software to run under TOPS-20 is to produce for each program a single executable module which can run under either operating system. This module must,

of course, determine the system on which it is executing and, where appropriate, execute slightly different code depending upon the system. The advantage of this approach over that of producing two executable modules, one for TENEX and another for TOPS-20, is that it greatly simplifies procedures for software maintenance and enhancement. To update a program, modifications to the program need to be entered only once and only a single new executable module needs to be created. We have examined the TENEX implementation of MSG and foresee no major obstacles in accomplishing such a conversion.

As mentioned above, we have begun to convert MSG to run under TOPS-20. There are two different aspects to this conversion process:

- The call/return conventions for several JSYSs used by MSG are different for TOPS-20 than for TENEX. References to these JSYSs by MSG must be modified to behave appropriately for the system being used.

- Certain TENEX features used by MSG are not supported by TOPS-20 and will not be supported even after TOPS-20 is enhanced for NSW. Where TOPS-20 provides functionally equivalent features MSG will have to be modified to use them. Where none exist MSG will have to be changed and its capabilities perhaps limited when it executes under TOPS-20. For example, MSG uses a TENEX inter-job communication facility which TOPS-20 does not support. However, TOPS-20 has an interprocess communication facility (called IPCF) which provides the inter-job communication functions required by MSG. MSG will be modified to use the IPCF primitives when running under TOPS-20.

MSG makes use of the TENEX JSYS trap mechanism as the means by which it gains control when a process executes a communication

primitive (e.g., SendSpecificMessage). TOPS-20 will be enhanced to support JSYS traps. However, the enhanced version of TOPS-20 will not be available for some time. Until JSYS traps become available, TOPS-20 MSG will make use of another more restricted mechanism for passing control between processes and itself.

As part of our efforts to measure and improve NSW system performance, we have instrumented MSG to measure the delays incurred as control passes from a process to MSG when a communication primitive is executed, and from MSG to a process when a pending event (e.g., outstanding receive operation) is completed. These delays are principally due to the amount of time required for the TENEX or TOPS-20 scheduler to stop running one process and start running another, and reflect both scheduler overhead and the number of other processes contending for the processor.

Making these measurements requires some cooperation between MSG and the process under its control. We implemented a pair of communicating processes which cooperate with MSG as required to make these measurements. Immediately before one of the processes executes an MSG primitive it reads the system clock and stores the time in one of its accumulators where it can be accessed by MSG. When MSG gains control after the primitive has been executed, it also reads the clock and can compute the delay incurred in gaining control from the process. A similar procedure is followed when MSG completes a pending event. Before

signalling the process that the event has completed, MSG reads the system clock and stores the time in one of the process accumulators. When the process resumes execution (an "unblock" signal is assumed) it subtracts the time supplied by MSG from the current time in order to obtain the delay incurred in gaining control from MSG.

Table 1 summarizes preliminary measurements made in May for these control-passing delays. These preliminary measurements indicate that in approximately 90% of the cases less than 100 ms are required to pass control between MSG and a process it is managing. However, in a small but significant number of cases considerably more time is required. We do not at present, understand why this is the case and we plan to study this question in more depth.

MSG supports a command language which can be used to to monitor and control both MSG and the processes MSG controls (See BBN Report No. 3450). A special control character (CNTL-S) is used to invoke the MSG command language interpreter. Use of CNTL-S for this purpose has caused some difficulties for NSW users. There are two related problems here. First, since MSG uses CNTL-S as a signal to activate its command language interpreter, NSW users cannot transmit CNTL-S to tools; and second, if a user should type CNTL-S, either accidentally or in an attempt to transmit it to a tool, he would find himself interacting directly with the MSG command language interpreter.

JSYS Trap Delays

Delay	%	Cumulative %
0-50	62.6	62.6
50-75	10.0	72.6
75-100	20.5	93.1
100-125	5.4	98.5
125-150	.7	99.2
150-200	.4	99.6
200-250	.2	99.8
250-500	.2	100.0
500-750	0.0	100.0
750-1000	0.0	100.0
>1000	.01	100+

Unblock Signal Delays

Delay	%	Cumulative %
0-50	56.7	56.7
50-75	24.0	80.7
75-100	6.0	86.7
100-125	2.9	89.6
125-150	1.4	91.0
150-200	5.7	96.7
200-250	.8	97.5
250-500	2.2	99.7
500-750	.2	99.9
750-1000	.05	99.95
>1000	.2	100+

TABLE 1

This generally causes much confusion. These problems have been corrected by deactivating CNTL-S for MSGs which are created by the NSW Dispatcher. Since all user Front End processes are managed by MSGs created by the Dispatcher, CNTL-S is an ordinary character for NSW users. However, since the MSGs which manage other system components are not created by the Dispatcher, CNTL-S can still be used by system implementers and operators to invoke the MSG command language interpreter.

A number of debuggers (DDT, IDDT, BDDT) are accessible through the MSG command language interpreter. NSW component implementers use these debuggers during component checkout and system integration. During this quarter we added a fourth debugger, called DAD (for Do-All Debugger). DAD is a powerful, multi-fork debugging aid developed by SRI. The ability to use it from within MSG should greatly facilitate installation of NLS as an NSW tool as well as ease the integration of the "SRI Front End" into NSW.

During this quarter a number of bugs in the TENEX implementation of MSG have been detected and corrected. One of the more interesting bugs was a synchronization problem that occurred during MSG initialization. An active MSG configuration consists of a collection of TENEX jobs started at initialization time by a "central" or "controlling" MSG job (See BBN Report 3450 for details). At initialization the controlling job creates, in sequence, the subsidiary jobs which directly control the various NSW system components (e.g., Foreman, File Package). The synchronization problem was that it was possible for a process (such as a Foreman) to attempt to send a (generically addressed) message to another process (such as a Works Manager) before the MSG job for the receiving process had been created and started by the controlling MSG. The result was that the message would be lost.

This problem first became evident during the checkout of the new NSW system that supports the interim reliability plan (See BBN Report No. 3751 and Section 3 of this report). As part of this reliability package a Foreman process is created whenever a tool bearing host restarts. This Foreman process checks for any tool sessions that may have been left uncompleted as the result of a crash. Part of its responsiblity is to send a message to a Works Manager process to notify it that the tool bearing host has been restarted. Whenever a host that supports both the Works Manager and NSW tools (i.e., a Foreman) was restarted, this Foreman message induced the race condition. Once detected, the problem was relatively easy to correct by properly synchronizing the subsidiary jobs during MSG initializaton.

At a meeting hosted by the Rand Corporation on the subject of interprocess communication for the joint ARPA/Navy ACCAT program MSG was adopted as the standard mechanism for interprocess communication within ACCAT. At present the ACCAT facility includes three different host operating systems: TENEX, TOPS-20, and UNIX. When we complete our conversion of MSG for TOPS-20 next quarter, MSG implementations will exist for two of these three systems. As part of our work on the ACCAT project (supported by another contract) we will be implementing MSG for the UNIX operating system.

Finally, we report that Computer Corporation of America is planning to use MSG to support the interprocess communication

requirements of SDD-1, a sophisticated data base management system for distributed data bases.

3. The TENEX Foreman

Last quarter we reported on the development of an instrumentation package for the TENEX Foreman component (see BBN Report number 3751). The probes for this instrumentation package, which are inserted at strategic locations directly into the Foreman code, generate raw binary data relating to message traffic, abnormal occurrences, and performance data. Only binary data is recorded in order to minimize the on-line processing time. The raw binary data for the Foreman sessions of a given NSW host incarnation is collected in a single file. Periodically, these files are retrieved and processed.

This quarter we have developed a software package to do the data reduction and formatting necessary to utilize the raw data, and to enable us to summarize large quantities of data. The data analysis program is capable of processing multiple raw data input files either in their entirety or on a session by session basis. It can automatically generate separate textual reports covering session summaries, message traffic, detected failures, and performance data. The session summaries report the name of the tool used, the time and date of the session, and identifiers to help locate the data for this session in other reports. The failure report summary provides a partial state description for all sessions that terminated in an abnormal fashion. This is generally sufficient to isolate the problem. Even when it is

not, it has served to alert us to the existence of the problem, enabling other measures to be taken to more accurately pinpoint the cause. The performance data summaries are currently oriented toward timing the intervals (both CPU time and real time) associated with the major Foreman functions. These include measuring the time to begin a tool session, the time to terminate a tool session, and the time to retrieve and replace NSW files. In addition to the various computed intervals, the performance summary includes data indicating the group and system load occurring at the time of the event. We are currently experimenting with the format of the performance summary in an effort to expedite its use as direct input to statistical analysis programs. During the next quarter we will be using the automatic data analysis program to process the data generated by the new Foreman releases.

At present, all TENEX programs which are NSW tools execute in an encapsulation mode. In this mode the Foreman intercepts (traps) certain operating system calls (JSYSs) made by these programs (such as file manipulating operations) before they can be acted upon by TENEX and transforms them into equivalent NSW operations. Refer to BBN Report No. 3266 for a more detailed discussion of the principles of encapsulation. To support this mode of operation the TENEX Foreman includes an encapsulator module which is invoked by the Foreman when a JSYS executed by a tool is trapped.

Ideally, encapsulation makes it possible for existing programs to be installed into NSW as tools without modification. In practice, the success of the encapsulation approach depends upon the functional commonality between the tool bearing host operating system and the NSW, and the degree various tools exercise tool bearing host functions not naturally mapped into NSW functions. Encapsulation has been fairly successful for TENEX hosts. However, it is not uncommon for the installation of a new TENEX tool to require minor enhancements to the encapsulator module.

The process of installing a TENEX program as a tool includes trying to run it under the Foreman to determine whether it behaves properly when its operating system calls are trapped and transformed by the encapsulator module. If the tool misbehaves, then the problem must be diagnosed and corrected. In some cases the encapsulator module must be modified to properly transform some of the JSYSs used by the program. In other cases the name of a non-NSW file used by the tool (e.g., the dictionary file used by SPELL) must be added to a list of local files referenced by the tool. This list is passed by the Works Manager to the Foreman when the tool is started. The Foreman allows direct access by the tool to all files on this list. Thoroughly exercising a program in this way is a very tedious procedure in a fully configured NSW system.

To facilitate the installation of TENEX programs as NSW tools we have developed and implemented a "test encapsulator" called WDYT (for Who Do You Trap). WDYT provides a more orderly means for examining the behavior of TENEX software being installed as NSW tools and can be used in a "stand alone" mode that does not require an NSW system.

The program to be installed as a tool runs under the control of WDYT. WDYT is set up to trap exactly the JSYSs trapped by the Foreman encapsulator module. When one of these JSYS traps occurs, the tool installer can use WDYT to examine the particular JSYS along with the parameters supplied by the tool. Given knowledge of the behavior of the Foreman encapsulator module, the tool installer can determine whether it will properly handle the JSYS. If not, he can decide whether modification to the encapsulator module, addition of a file name to the tool's local file list, or some other modification is required. Initial experience with WDYT indicates that it significantly reduces both the amount of time required to install a tool and the likelihood that the tool will misbehave in the NSW environment after it has been installed.

During this quarter we installed several TENEX software packages as NSW tools including SPELL, LINKER, BCPL, BDDT, and JIGSAW. SPELL is an interactive spelling corrector developed at Stanford and enhanced at BBN. It accepts as input a text file and checks the spelling of each word in it. For words that it

believes may be misspelled, it informs the user and gives him several options. These include: keeping the word as currently spelled, correcting the word as recommended by SPELL, selecting one from a possible set of correctly spelled words supplied by SPELL, or replacing the word with one supplied by the user. The previously installed text editing tools, together with MRUNOFF and SPELL, form the basis for a powerful documentation preparation facility within NSW.

LINKER is the NSW name for the DEC PDP-10 linking loader, LINK10, which is used to produce executable object modules (core images) from relocatable binary files (.REL files). When used under TENEX, LINK10 terminates after it completes construction of a core image from the specified .REL files. The assumption is that the user will use the SAVE or SSAVE command of the TENEX EXEC to save the core image in a file. Since the NSW command language does not have the concept of saving a core image, LINK10 had to be augmented for the NSW environment to save the core image in an NSW file. This enhancement to LINK10 was done in a modular way so that any new DEC releases of LINK10 can be installed simply by ensuring that LINKER uses the new version of LINK10 rather than the old one.

BCPL is the compiler for the BCPL programming language. It produces .REL files from BCPL source programs. BDDT is a debugging package for the BCPL language. It allows users to debug BCPL programs in terms of source language constructs (such

as statements, subroutine calls, structures, and variables) rather than machine language concepts (such as instructions and memory locations). With the installation of BCPL, BDDT, and LINKER, NSW now supports the edit/compile/debug cycle for the BCPL language. This is a significant achievement because, in principle at least, it is now possible for NSW to support the development of those NSW system components which are implemented in BCPL. These include the Works Manager and the Front End.

JIGSAW is a preprocessor for the JOVIAL programming language. It represents the first JOVIAL tool installed in NSW, and as such is the first step toward developing a JOVIAL programming environment within NSW. JIGSAW was installed at the request of Rome Air Development Center. To install it, we obtained the FORTRAN source programs for it from TRW. We created a TENEX core image for JOVIAL by compiling these programs, after slight modification, using the TENEX FORTRAN compiler and loading the resulting .REL files using LINK10. This produced a JIGSAW for TENEX which was then installed as an NSW tool in the normal way. The TENEX JIGSAW operates in an interactive mode. JIGSAW is also being installed on a Multics tool bearing host through a GCOS simulator for use in batch mode. This nicely illustrates the potential flexibility in selecting an appropriate tool set, which is a major goal of NSW.

Part of the reliability plan for the NSW requires that the Foreman maintain tool workspaces in a crash proof manner. This

is to ensure that files that may have been trapped in a workspace as the result of a system crash occurring before the normal termination of a tool session are not lost. It is the responsibility of the Foreman to maintain these workspaces until the user has an opportunity to retrieve files he is interested in by "rerunning" the tool. The files saved for the tool sessions are currently maintained in place within the workspace. See BBN Report No. 3751 for a more complete discussion of this.

For the TENEX Foreman this poses a potential problem. The problem derives from the fact that on TENEX workspaces are implemented by a fixed number of file directories. If enough users are slow in retrieving files from saved workspaces it is possible for all of the workspaces to be used for saving incomplete tool sessions, thereby preventing new tool sessions from being initiated. One solution to this problem is to implement a "two level" workspace scheme. This would allow saved workspaces to be moved from workspace directories to a secondary storage area, releasing the workspaces for use in new tool sessions. When a user attempts to retrieve files from a saved workspace that had been moved to secondary storage, the saved workspace would first be returned to a workspace directory, after which the file retrieval would occur as usual. During this quarter we designed a two level workspace scheme. The design is presented in Appendix A of this report.

Finally, during this quarter we released and integrated into the NSW system several new versions of the TENEX Foreman. In addition to minor bug fixes, these releases included a number of changes and improvements, some of which are described in the paragraphs below.

Due to recent changes in the NSW inter-component protocols the Foreman must be prepared to receive and react to asynchronous messages from other components. By an "asynchronous message" we mean here a message which is not part of an on-going protocol exchange between the Foreman and some other component; for example, a message from the Front End to terminate a tool session. In the earlier versions of these protocols, in those situations where the Foreman had to handle such asynchronous messages it would be alerted by an MSG alarm to issue a receive. In the new protocols the alarm is often omitted. This required the Foreman to be changed so that it always has an outstanding receive operation. Since the Foreman, in general, will be busy monitoring its tool's execution while this receive is pending, it requests a PSI (rather than Unblock) completion signal from MSG for the receive.

Under certain circumstances, such as a termination request from a tool, the Foreman enters into a more sequential mode of operation in which it also expects to receive a message. Since a receive is already outstanding, in principle the Foreman need not issue another one. However, in these situations the Foreman

prefers an Unblock completion signal to the PSI signal associated with the pending receive. Because MSG, at present, does not allow a process to change the signal associated with an outstanding operation, the Foreman must rescind the pending receive and then issue a new one with the desired completion signal. We are considering adding a "change signal" operation to MSG to allow simplication of process actions in such situations.

If the Foreman is engaged in a protocol exchange with another component, such as one to open a file, when it receives certain asynchronous messages, such as a request to terminate tool execution, it must abandon the on going exchange and enter into a new one with the sending component. To do this correctly, the Foreman must be able to determine the protocol exchange subsequent messages belong to so that it can ignore messages from the abandoned exchange and process messages from the new one. Early versions of the Foreman did not do this properly. The current Foreman uses TIDs (Transaction IDs) as a means to distinguish among these messages.

When a tool is started, the Foreman establishes a communication path with the Front End for the tool. For early versions of the NSW all the interactive tools normally communicated with the user directly through his terminal. Consequently the Foreman always established TELNET connections with the Front End. In order to accommodate NLS and other tools which are designed to interact with the Front End component in

more general ways than TELNET connections can easily support, the Foreman has been modified to additionally support binary MSG direct connections. The type of connection required by a particular tool is stored in its tool descriptor maintained by the Works Manager and passed to the Foreman as part of the RUNTOOL protocol scenario. The Foreman uses this information to cooperate with the Front End to establish the proper communcation support.

The Foreman has been changed to be more selective in the way it uses the Front End help facility in file open operations. In the message it sends to the Works Manager to open a file on behalf of a tool the Foreman must specify the action to be taken by the Works Manager if the file specification supplied by the tool is ambiguous. The Foreman has three options: the Works Manager should consult the user via the Front End help facility to disambiguate the file name; the Works Manager should consult the Foreman to do the disambiguation; or, the Works Manager should tell the Foreman that the open failed. The current Foreman uses a heuristic to determine the origin of the file name in an attempt to select the appropriate option. If it believes the file name was one supplied by the user (e.g., the name of a file to be edited), it instructs the Works Manager to consult the Front End for help. If it believes the file name was one generated internally by the tool, it instructs the Works Manager to report failure if the name is ambiguous. The reasoning here is that many tools are equipped to handle such failures; for

example, certain text editors look for files containing mode settings but are prepared to continue operating if they can't find them. Additionally, it was felt that it would be very confusing to a user if he were asked to disambiguate a file name for an open operation he had no knowledge was even occurring. This procedure is heuristic in the sense that it is not possible to correctly determine the origin on the file name in all cases. However, we have been successful within the limited tool set currently available through NSW.

Appendix A

The Management of Saved Tool Sessions
Foreman Extensions Enabling Workspace and Tool Session Recovery

The present TENEX Foreman implementation of the NSW interim reliability scenarios preserves the state of the local filespace of the running tool by reserving the TENEX directory assigned as the tool's workspace. Since only a finite number of workspaces are defined for a given NSW tool-bearing host, a sequence of events which results in the preservation of a number of local name directories (LND's) rapidly depletes the stock of available workspaces. It is desirable, then, to provide a more satisfactory mechanism for the preservation of working files from the time of a SAVELND operation to the occasion of restarting the tool session or delivering the files back to the global NSW file system.

This note describes the present implementation of the LNDSAVE scenario in the Foreman component and defines the modifications which are proposed to relieve the problem of a diminishing supply of workspaces.

1. Present Implementation

There are several circumstances under which the Foreman may discover that a particular tool session can no longer be continued. We distinguish two cases, which will be termed individual and general. In the individual case, a specific event occurs such as the timeout of a FE or WM function, the loss of the tool-FE connection (and failure to reestablish), or the receipt of the message FM-SAVELND. In the general case, the Foreman is presumed to be starting up after the TBH is restarted, either due to a crash or a normal shutdown; routine examination of the state of the current workspaces discloses the presence of local working files undelivered at the time of shutdown. In an individual case, the specifically interrupted tool session cannot be continued. In the general case, none of the tool sessions active at the time of Foreman shutdown can be resumed without both user presence and interaction with other NSW components.

In either case, a number of specific actions are performed for protection of the files associated with the tool sessions which cannot be continued. The single set of actions which concern us in this discussion are implemented through the subroutine RESCUE for the individual case and FORSAV for the general case. Since the functions of FORSAV are partially determined by antecedent functions of RESCUE, that routine will be discussed first.

As an introduction to the following discussion, it must be explained that, on TENEX, each NSW tool executes as an inferior process of a Foreman connected to a private directory, hereafter referred to as a "workspace". The TBH provides a finite set of workspaces for the use of NSW tools. Each time a tool is started, its controlling Foreman selects a workspace from a shared data base and connects the tool fork to the corresponding directory. All TENEX Foremen associated with a given NSW TBH and incarnation run in individual forks which are initially connected to a common directory containing the Foreman data base file, FORCOMFILE.SHR. The common data base file contains two data structures--the workspace list (or the WE. structure) and the rerun list (or the RR. structure). In addition, the common file contains a semaphore by which a Foreman can obtain exclusive access to the common data base.

Each entry in the workspace list comprises three words, arranged as follows:

```
WE.USE,,WE.DIRNO
WE.XTRA,,WE.PSWDPTR
WE.TOOLID
```

WE.USE always contains a zero if the workspace is not in use, and the NSW process instance number of the controlling Foreman if currently active. A workspace not under Foreman control may be blocked from reuse or deallocation by setting WE.USE to -1 (777777). WE.DIRNO is the number of the workspace directory, WE.PSWDPTR is an optional to the password string for this directory, and WE.TOOLID the unique tool-id assigned to this tool session by the Works Manager. WE.XTRA is an unused half-word. There is one and only one entry in the WE. structure for each available workspace.

Each entry in the rerun list comprises two words, arranged as follows:

```
RR.ITYP,,RR.IDX
RR.WMFLAG
```

RR.IDTYP always contains a zero if the rerunlist entry is not in use, and a +1 if it is in use to record a rerunnable tool session. RR.IDX is an index to the WE. structure entry for the

workspace containing the rerunnable tool session. RR.WMFLAG is always zero if the tool session has been saved but not yet acknowledged as rerunnable by the Works Manager. When the Works Manager finally acknowledges the tool session as rerunnable, RR.WMFLAG is set to the current TENEX day and date obtained from the GTAD JSYS (if none is available, -1 is used instead). There is one and only one entry in the rerun list for each workspace containing a rerunnable tool session.

Whenever a condition is detected in the Foreman which requires the preservation of a tool session, the routine RESCUE can be used to protect the workspace and create the appropriate rerun list entries. A separate routine, FORSAV, exists to process the entire domain of available workspaces when the NSW on the TBH is restarted.

1.1 Preservation of Indivdual Tool Session State

When an individual tool session must be preserved, the Foreman stops the tool, closes any open files, and examines the LND (via the routine WSUSED) to see if it contains any entries. If there is at least one entry in the LND, we assume the tool session should be preserved. A special code (777777) is entered in location WE.USE assigned to the tool's workspace in the Foreman common data base, and the index to WE. which identifies the workspace is entered in a location RR.IDX in the rerun list. RR.WMFLAG. RR.IDTYP is set to 1, to indicate an entry in use to address the WE. structure. RR.WMFLAG will be set to the time and date when the NSW Works Manager confirms the rerunnability of the tool session preserved in the workspace indicated by RR.IDX. When RR.WMFLAG contains a zero (and RR.IDTYP = 1) the tool session has been identified as rerunnable by the Foreman, but this state has not yet been confirmed by the WM.

It is necessary to modify this strategy to permit the files contained in the workspace to be transferred (along with the LND file itself) to some appropriated holding directory, and to enable recovery of these files in a timely and efficient manner.

1.2 Preservation of Multiple Tool Session States

When the NSW Foreman job is started on a TBH, the Foreman MSG signals the first Foreman instance to perform initialization for all Foremen. Among other things, this entails the examination of each available workspace to determine if any are holding tool sessions interrupted at the time of a TBH crash. These are workspaces which are marked in the common page (location WE.USE) with a positive number, the instance number of the controlling Foreman at the time of the crash. For each such workspace, the rescue operation outlined above is performed.

This examination also picks up any workspaces locked by RESCUE (WE.USE = 777777) but not entered on the rerun list (no matching RR.IDX entry), just in case the TBH crashed while performing an LNDSAVE.

After the initializing Foreman updates the rerun list and corrects any omissions detected, the old saved tool sessions are examined and discarded if their RR.WMFLAG entries indicate the session was saved longer ago than HLDTIM (currently 4 weeks). (It is always possible for the WM host to crash before the tool session salvation has been recorded in its data base, but after the Foreman has been notified, leaving the Foreman holding a rerunnable tool session not recorded in the WM data base. This can be overcome by reporting rerunnable tool sessions one or more additional times, even if WM confirmation of rerunnability has already been received.) Routine FMWMK2 is called to send the WM-LND-SAVED message to the WM, reporting all saved tool sessions not previously acknowledged by the WM. On reply from the WM, the routine FMOKREP marks RR.WMFLAG with the time and date for each tool session confirmed as rerunnable by the WM. Tool sessions reported from the WM as non-rerunnable have their workspaces freed (WE.USE <== Ø) and their rerun list entries deleted (RR.IDTYP <== Ø; RR.IDX <== Ø; RR.WMFLAG <== -1 (in the future the senses of Ø and -1 for this flag will be exchanged)). The Foreman common page is kept locked while awaiting the WM reply to WM-LND-SAVED. Like all interactions with the FE or WM which require a reply for continuation of a specific action, the receive specific issued to MSG for this transaction sets a timer on transmission of this message so that appropriate remedial action can be taken if the reply is not promptly received. This provision assures that the Foreman common page will not be left locked for lengthy periods of time. The common page lock is automatically cleared during startup after a crash of the TBH. In all interactions between the WM and the Foreman, the tool sessions are identified by "tool-id". The value of the tool-id for each session is saved in the location WE.TOOLID in the common page. The WM is alleged to assure non-duplication of values of tool-id for a given series of NSW incarnations.

1.3 Resumption of Suspended Tool Sessions

The Foreman attempts to resume a previously suspended tool session when it receives the FM-REBEGINTOOL message. Routine FNDWSFCN is used to retrieve the correct workspace (corresponding to the tool-id sent by the WM). FNDWSFCN also assigns tool-id's to unused workspaces when invoked to handle the FM-BEGINTOOL message. At present, no check is made to determine if the tool-id of a new FM-BEGINTOOL request is already in use for a previously saved tool session. Since the WM is assuring non-duplication over extremely long time periods, this should not

be problem. If, however, a duplication should arise, there is a probability that files saved for the user of the former tool-id might be delivered to the new user, and/or vice-versa. Recovery of an old workspace is accomplished through routine GOLDWS, which searches the rerun list for entries into the workspace table (RR.IDX, when RR.IDTYP = 1). If WE.USE = 777777 for the entry indicated by RR.IDX, WE.TOOLID is tested against the tool-id value received from the WM. A match, of course, indicates that the workspace containing the desired saved tool session has been found. The workspace is assigned to the current Foreman instance (WE.USE <== MYINSTNO) and the corresponding entry in the rerun list is freed for reuse (RR.IDTYP <== Ø; RR.IDX <== Ø; RR.WMFLAG <== -1). The Foreman connects its fork to the designated directory and enters the regular code to wait for further instructions (STARTOOL, TERMINATE, ABORT, etc.), after replying to the WM.

2. Planned Implementation

As noted previously, while workspaces do not represent a costly TENEX resource, there is still only be a limited number of them available to a given NSW TBH at any one time; therefore, since there is no practical limit to the number of interrupted tool sessions which may have to be saved, it is desirable to provide a way to save the various local files associated with an interrupted tool session without using one of the available workspaces.

To accomplish this, certain additional conventions will be adopted. A special directory will be required on every TENEX TBH to hold files salvaged from interrupted tool sessions. The name of this directory will be saved in the common page and initialized through the MKCOM program. The directory name will be found in a block labeled SVDRNM. The entire contents of the LND and its files (i.e., not any TENEX temporary files) associated with an interrupted tool session will be transferred to <svdrnm> at some appropriate time after detection of the interruption of NSW function. By preserving the entire state of the directory associated with the tool session, it may be possible to resume the tool session with the files in approximately the same state as when the interruption initiating the LNDSAVE occurred. This should make the tool sessions appear to be relatively continuous in the presence of short-term failures of various NSW components. Obviously, the conditions at the time of the failure cannot be exactly duplicated. It will be left to the user's discretion as to whether a tool should be continued in this approximate state or should merely have the pertinent files delivered for later use.

The WM-assigned tool-id will become the major identifier for the recovery of interrupted tool sessions; a new entry, RR.TOOLID, will be added to the rerun list data structure. The entry WE.TOOLID will be preserved, since some source record of the tool session in progress will be needed for TBH crash recovery. A large number of new states will be needed for RR.IDTYP, which will be the principal indicator of status for each saved tool session. State +1 of RR.IDTYP will be used to indicate a transitory state when a tool session has been identified as savable, but the WM has not yet acknowledged it as such. State +2 will connote a rerunnable tool session which is saved in a workspace, rather than in <svdrnm>. Saving a tool session in a workspace is an efficiency measure by which, in selected cases, we hope to avoid the overhead of first moving all of the files out of the workspace and then moving them back into (perhaps) another workspace when the tool is continued. Our design is based on the philosophy that we should be able to remove the LND from a workspace at any time, if it becomes prudent to do so. State +3 of RR.IDTYP will be used to indicate a transitory state during the time local files are being moved from the workspace to <svdrnm>. RR.IDTYP with state value +4 means that the saved tool session is recorded in <svdrnm>. State +5 will be a transitory state during the time files are being moved from <svdrnm> into a new workspace from which the tool will be rebegun. State +6 will be a transitory state indicating that a tool session exists, but contains no savable files, pending WM notification of this fact. State +7 indicates a tool session which is to be discarded by garbage collection. Examination of the state variable will facilitate recovery if a TBH crash occurs during the general LNDSAVE accompanying Foreman initialization. RR.IDX will be undefined when RR.IDTYP = +3, +4, or +5.

A naming convention for local files will be adopted which simplifies identification of all files associated with a particular tool session when stored in <svdrnm>, while assuring non-duplication of file names. As a further policy, we will also adopt this convention within the confines of the workspace so that each file has a single, local name throughout its existence, even when moved from a workspace to <svdrnm> and back to a workspace. This convention greatly simplifies the procedures involved in saving a workspace. Duplication protection could be further assured by modifying routine WSGET, which assigns new free workspaces, to search the rerun list for duplications of the tool-id received from the WM; however, this appears to create an unreasonable burden of verification for the Foreman. Accordingly, this modification will not be implemented. Instead, we will rely on the WM to make a unique assignment of tool-id. (To facilitate the recovery of specific tool-id's from the rerun list, the RR. structure will be expanded in size, and the entry selection in routine LWSFRR will be converted to a hashed-address scheme. This accessing scheme will be propagated throughout the rerun list manipulating routines wherever appropriate, and GOLDWS

will be modified to search for tool-id from the RR. structure, rather than the WE. table, as in the current implementation. These modifications are discussed in detail below.) Every local file name, whether in a workspace or <svdrnm>, will begin with an arbitrary string, such as "XNSW-", followed by the WM-assigned tool-id as an ASCII string giving a 12-digit octal value. The name of all local files associated with a given tool session will be distinguished by its extension. The extension will be an ASCII string giving a 6-digit octal value equal to the present NSW incarnation number obtained by the Foreman instance from MSG, concatenated with a 12-digit octal value representing the time (in milliseconds, at file creation), since the TENEX system was restarted. The value of the duration of TENEX service is obtained via the TIME JSYS.

We believe this approach will assure that two files are never assigned the same name. Unless the WM makes an error, each tool will use a unique tool-id for naming its files. Each new NSW file name created by a tool requires trapping to the Foreman. In general, since it takes significantly more than one millisecond of real time to execute the appropriate Foreman code, the system time read by the TIME JSYS will be different for each instance of file creation, and each new file created will have a unique name. It is possible that at some future time there will exist NSW tools which run in multiple forks, making it possible for two or more forks to simultaneously attempt to create NSW files. Redundant naming in this case will be precluded by establishing an internal Foreman register to record the system time last used in naming a file. This register will be accessed only after testing and setting a mutual exclusion semaphore. If the time read by the TIME JSYS is less than or equal to the stored time, the stored time will be incremented and the incremented value used as the basis for naming the file being created. We believe it will then be possible for the Foreman to assure the uniqueness of the extension field of the file name under the following assumptions.

Although the system time clock cycles once in an average of 2 years, 61.9 days, it is extremely unlikely that a given TENEX TBH and/or NSW incarnation will remain continuously active for that period. Since restarting the TBH generally implies a system clock reset, there is the possibility of having the same system time field string for a subsequent file creation (P = .000116 for a TBH creating 10000 files between resets every 24 hours). To make sure there will be no duplication, the TENEX NSW incarnation number, which is incremented each time the central MSG job is started, will be used to further identify files. Successive version numbers for subsequent versions of the NSW files will be assigned under normal TENEX conventions.

The file naming conventions will require modification of the name-generating program in the encapsulation code, as well as

renaming of all files delivered to the Fo eman by the File Package. The LND file will be named slightly differently to facilitate identification. The extension will be "LND", and the principal file name will be simply the 12-character ASCII/octal tool-id, without the "XNSW-" prefix. The LND is named differently from all other files of a given tool session both to identify it and to permit the GNJFN JSYS to be used to reference the family of non-LND files for the session. Thus, with these conventions, it is easy for a program (or operator) to reference the family of all saved LND files, or the family of all saved files for any particular tool session.

An example: A tool session has been started by Foreman 1106, running in NSW incarnation 1776, with tool-id 453271602413 in workspace FWK7. Its LND file is named:

<FWK7>453271602413.LND;1

After some period of operation, some files have been created. Three versions exist of a file begun at "112345047654" and one version of a file begun at "112345047773". The files associated with this tool session are:

<FWK7>XNSW-453271602413.001776112345047654;1,2,3
<FWK7>XNSW-453271602413.001776112345047773;1

When an LNDSAVE is performed for this tool session, FWK7 will be freed for reuse after copying these files to <svdrnm> as:

<svdrnm>453271602413.LND;1
<svdrnm>XNSW-453271602413.001776112345047654;1,2,3
<svdrnm>XNSW-453271602413.001776112345047773;1

Note that no attempt is made to determine which files or versions will eventually need to be delivered to the file system.

The following sections describe in greater detail how these new conventions will be used in implementing the transfer of files to and from <svdrnm> in the context of the interim reliability scenario implementation described above.

2.1 Individual LNDSAVE

Under the implementation plan, the salvation of individual tool sessions proceeds as described in 1.1 above--the tool is stopped, the files are closed, and the workspace is marked (WE.USE <== 777777) and entered on the rerun list. The rerun list entry used by routine LWSFRR will be selected by converting the tool-id to a 9-bit hash index to RR. Search for a free entry in RR. will proceed from this starting point. When a free entry is located, WE.TOOLID will be copied into RR.TOOLID, RR.IDTYP

will be set to +1 (indicating temporary preservation of the tool session with files in the workspace), RR.WMFLAG will be set to -1 (indicating that a tool session has been saved, but not yet confirmed as rerunnable by the WM), and RR.IDX will be set to the index of the workspace holding the saved tool session in the WE. structure. The sequence of these operations is important. It is necessary to insure that the Foreman making a rerun list entry can complete the entry without competition from other Foremen; therefore, the operation begins with the Foreman testing and setting the mutual exclusion semaphore for the Foreman common page.

Marking WE.USE has first priority, since it blocks deallocation of the workspace by all other Foreman processes. Loss of an incomplete rerun list entry is of no importance, so RR.TOOLID and RR.IDX can be completed next. RR.WMFLAG can be set to -1 at this time. Finally, setting RR.IDTYP to +1 signals that a stable state has been reached from which the tool may be rebegun. (Note that prior to this, RR.IDTYP has necessarily been zero, since this is the only condition under which a rerun list entry can be considered to be free.)

A special state, RR.IDTYP = +6 is reserved to indicate that the LND for the saved workspace contains no files. For such tool sessions, the Foreman will attempt to notify the Works Manager that it is discarding the entry. After setting RR.IDTYP to +1, the LND is examined to determine if it contains any files. If none are found, RR.IDTYP is changed to +6. At this time the mutual exclusion semaphore on the Foreman common page may be freed to allow access by other Foremen.

The tool cannot be rebegun until the WM has acknowledged the tool session as rerunnable. RESCUE will be modified to check the LND to see if it contains any files. The WM-LND-SAVED message will be generated whether or not the LND contains any files; however, if the LND contains no files, RR.IDTYP will have been set to the special state +6 and a NEW list argument in the WM-LND-SAVED message, "empty-tool-session-list", will be sent with a single element equal to the tool-id. If a non-empty LND was previously found (RR.IDTYP = 1), "empty-tool-session-list" will be returned with no elements and the tool-id will be reported in "saved-tool-session-list". (This is a proposed modification to the present protocol, and requires the concurrence of COMPASS and other NSW participants.) The WM will reply by returning the tool-id on the "exception-list" if either the tool session cannot be identified, or it is OK to throw it away. If the tool session is to be discarded, WE.USE is immediately cleared, along with RR.IDTYP, on receipt of the reply to the message WM-LND-SAVED. Then, the other rerun list entries may be cleaned up (RR.WMFLAG <== Ø, etc.). If the tool session is to be saved, it is only necessary to set RR.WMFLAG to the date

and time reported by the GTAD JSYS, as previously described. The LNDSAVE is completed by changing RR.IDTYP to state +2. It is necessary for these modifications to the data base to be performed under the protection of the common page mutual exclusion semaphore, since resetting RR.IDTYP will permit another Foreman to attempt to use the rerun list entry being reset. The need for mutual exclusion can be avoided by making RR.IDTYP the last rerun list entry to be changed.

This stable state (files preserved in protected workspace, RR.IDTYP = +2) is roughly equivalent to the state in which all tool sessions are preserved under the present implementation. As mentioned previously, we want to eventually move the files from this protected workspace to the general directory <svdrnm>. There are many strategies by which this could be accomplished. We could agree to start moving the files as soon as the WM has confirmed rerunnability; however, since many LNDSAVEs will be in response to transient failures of User-FE or perhaps FE-Tool connections, and since the user will probably want to continue his tool session as soon as he can get back into NSW, it would be more efficient to keep the files in the rerunnable workspace until the workspace is needed for a new tool session. Since the tool session which has just been LNDSAVEd is a current session, and since usually the best candidate to move to <svdrnm> is one of the older (if not the oldest) saved tool sessions, it is resonable to assume that no further manipulations will be immediately required on the session just saved. We will address the problem of when to actually move the saved files to <svdrnm> in the broader context of how best to manage shared TBH resources.

2.2 General LNDSAVE

In the case of a TBH restart, initialization and crash recovery proceeds essentially as outlined in 1.2 above. The mutual exclusion semaphore of the Foreman common page is locked, whether or not the common page was previously in use (only the initializing Foreman is permitted this liberty to gain control of the data base for restart). The initializing Foreman looks for workspaces which were in use at the time of the crash (indicated by 777777 > WE.USE > Ø). Each such workspace is initially listed on the rerun list. The LND is examined before listing each workspace; if the tool session has any files (and hence should be rerun), the rerunlist entry is made with RR.IDTYP = +1, but if the session has no files (and hence should be discarded), the rerunlist entry is made with RR.IDTYP = +6 and the workspace is deallocated. It is important at this point to check for the possibility of a TBH crash occurring during a prior LNDSAVE (either individual or general). This is disclosed by the detection of a protected workspace (WE.USE = 777777) which is not on the rerun list, or which is listed, but does not have RR.IDTYP

= +2 or +3. This crash state is processed quite simply. It is only necessary to assure that the states of the two data structures, WE. and RR. are properly synchronized before transmission of the WM-LND-SAVED message. Any locked workspace not on the rerun list is added to it, just as if the Foreman had come up with the workspace in use from a previous crash. It is probably easiest to simply expunge the old rerun list entry if it is incomplete, and create a new one; however, the logic by which sub-entries are made should assure that one can recover by "filling in the blanks". That is, if RR.IDTYP = +6, it is not necessary to recheck the LND. On completion of this initial phase of the general LND saving operation, every workspace in use must be locked and listed on the rerun list with RR.IDTYP = +1, +2, +3, or +5. All other rerun list entries must be in state +4, +5, +6, or +7. Eventually, in our recovery scenario, state +3 will be resolved to +2 and state +5 to +4, while states +6 and +7 require no special action since they indicate pending deletions. However, these actions need not be implemented as part of the initial steps of rescuing tool sessions in progress at the time of the crash.

The next step is to report all newly-saved tool-id's to the WM. The rerunlist is scanned and all tool-id's with RR.IDTYP = +1 are entered on "saved-tool-session-list", while those with RR.IDTYP = +6 are entered on "empty-tool-session-list". There should be no tool-id which has RR.IDTYP = +1 and RR.WMFLAG = Ø. These two lists are sent to the WM in the general WM-LND-SAVED message. The reply from the WM contains two lists, "acknowledged-tool-sessions-list" and "exception-list". All workspaces appearing on the exception list are freed and removed from the rerun list, providing RR.IDTYP = +1, +2 or +6. All others on the exception list are set to a "garbage state", +7. This permits the WM to delete tool sessions that have been previously saved. All workspaces appearing on the acknowledged list are time-stamped in RR.WMFLAG and then have RR.IDTYP set to +2 from +1. Confirmation of any other listed tool-id (RR.IDTYP not +1) is ignored. A non-existent tool-id will log a non-fatal error message.

This completes the initial recovery phases of the general LNDSAVE operation. The initializing Foreman releases its lock on the common page, enabling other Foremen to begin running or rebeginning tools. The initializing Foreman may now take the actions necessary to resolve rerun list entries with RR.IDTYP = +3 or +5 at the time of the restart and then enter a background service mode associated with continuing management of the TBH Foreman activities, including movement of files from the workspaces to <svdrnm>.

Unfortunately, it is possible for the TBH to have crashed with files being moved to or from <svdrnm>. If files were being moved from the workspace to <svdrnm>, RR.IDTYP will be +3, and

the proper recovery procedure is to delete and expunge all files in <svdrnm> associated with the rerunnable tool-id and then reset the session state to +2. A curious ambiguous state of the data base exists if the TBH crashed after all files had been moved to <svdrnm> and the workspace had been deallocated, but before RR.IDTYP could be changed to state +4. To allow for this contingency, a rerun list entry trapped in state +3 at TBH restart must be checked for the continued presence of a locked workspace; if one is found, then the file movement was incomplete and should be deleted in anticipation of a later try. If one is not found, then the file movement must have been complete, and the state of the rerun list entry should be changed to +4 instead.

If files were being moved from <svdrnm> to a newly assigned workspace, RR.IDTYP will be +5, and the proper recovery procedure is to deallocate the new workspace, if one had been assigned at the time of the crash, fix up the rerun list entry, and reset the state to +4. To assure that the new workspace, which is not rebeginnable, is deleted rather than processed by the LNDSAVE section of the Foreman restart procedures, the REBEGINTOOL implementation must change RR.IDTYP from +4 to +5 before it assigns the new workspace to rebeginning the saved tool session. There is no set of conditions under which the contents of the new workspace would be preserved. The REBEGINTOOL implementation must change RR.IDTYP from +5 to +2 only after all files have been successfully copied (including the LND) from <svdrnm> to the new workspace.

It is undesirable for a different Foreman to attempt to rebegin a tool session while the initializing Foreman is releasing it from one of these two intermediate states (+3 or +5). It is also undesirable to lock out all other Foremen from beginning or rebeginning any other tool for the duration of this cleanup operation. Accordingly, our implementation provides that any Foreman attempting to rebegin a tool session in state +3 or +5 will dismiss action for a period of time and try again later on. Since the cleanup of these intermediate states occurs immediately after the lock is released on the common page, there will be a minimum time before a tool session which is in one of these state can be rebegun.

After completion of all these actions, the initializing Foreman will enter a new mode of operation as a background service program, responsible for garbage collection of <svdrnm>, movement of files from old saved workspaces to <svdrnm>, and a number of other functions not directly related to the subject of this note. The detailed operational strategy for the background server is described in section 2.4 below.

2.3 FM-REBEGINTOOL Implementation Plan

As described in 1.3 above, resumption of a previously suspended tool session is initiated by receipt of the FM-REBEGINTOOL message. The message specifies the unique tool-id of the tool session to be rebegun. The rerun list is examined to find an entry with a RR.TOOLID matching the desired tool-id. When it is found, the associated RR.IDTYP is examined. If this indicates the tool session is in state +2, it may be rebegun immediately; the workspace is located via RR.IDX, the corresponding WE.TOOLID is checked, WE.USE is set to the current Foreman instance number, the rerun list entries are cleared, and the Foreman rebegintool code executed (this is essentially the same as the present implementation). If RR.IDTYP is Ø, +1, +6, or +7, this rerun list entry is not valid for rebeginning a tool session, and further action is required. For states Ø, +1, and +6, this is simply the reporting of an error to the WM; however, the function of the background server program is such that a tool-id may be listed twice, providing that one entry is +7. This state of affair occurs if the TBH crashed during execution of a previous REBEGINTOOL, after files had been moved from <svdrnm> to a workspace, but before the background server had garbage collected the archival entries of files in <svdrnm>. the search for matching tool-id must continue. If the state is indicated as +3 or +5, this entry is rebeginnable, but temporarily locked; the Foreman will dismiss for a while and try again (by which time the initializing Foreman process should have resolved the lock).

If RR.IDTYP indicates state +4, the tool session has been transferred to <svdrnm>, so the Foreman will execute code (FILGET) to retrieve the necessary files. This works in the following way. RR.IDTYP is set to +5 to indicate that the tool session is being retrieved. A free workspace is selected and assigned to the current Foreman. (WE.USE <== MYINCNO; WE.TOOLID <== RR.TOOLID; RR.IDX <== workspace-index) The index to this workspace and the tool-id are passed to FILGET, which maps the saved files for the tool session from <svdrnm> to the assigned workspace. The LND will be the last file mapped. After successful mapping of all working files and the LND, the Foreman completes the assignment of the workspace to the tool session by flagging the rerun list entry for garbage collection by setting RR.IDTYP to state +7. After the workspace has been set up with the old saved files, and the tool session flagged on the rerun list, the Foreman proceeds as before to connect its fork to the designated directory and to enter the regular code for file delivery and termination, after replying to the WM.

The new routine FILGET operates in the reverse manner from the component of the background server charged with moving files to <svdrnm>. FILGET first copies all files named "XNSW-(tool-id)" from <svdrnm> to the workspace and then copies

the LND. A successful return from FILGET causes release of the rerun list entry. For reasons discussed in 2.4 below, resynchronization in the event of a TBH crash can be accomplished fairly easily.

2.4 Background Server Functions

The initializing Foreman will support a number of background functions related to the ongoing management of TBH resources not directly related to any individual tool session. These functions include general management of the rerun list, the rerunnable workspaces, and the directory <svdrnm> for all functions not already assigned to the regular Foreman initialization and reliability code. On completion of the actions described in section 2.2 above, the initializing Foreman will enter the background service mode of operation. In this mode, the initializing Foreman (background server) will be responsive to messages via MSG and will respond to periodic signals of definite elapsed time periods to perform its housekeeping tasks.

Periodically, according to an algorithm discussed below, the background server will perform its routine processing of the rerun list. On these cycles, the Foreman common page will be locked only for brief periods of time, and only tool sessions in states +2, +4, and +7 will be examined. Every tool session in state +7 is to be expunged. If it has any files in <svdrnm>, those files are deleted and expunged. When these actions are complete, the rerun list entry is reset to the free state. These actions represent the garbage collection function.

Tool sessions in states +2 and +4 are screened according to their ages, as measured by the time elapsed since the recorded RR.WMFLAG time stamp. A state +4 tool session more than a given number of days old is simply marked for garbage collection (switch to state +7). A state +2 tool session more than a given number of minutes old may be transferred to <svdrnm>. To do this, the state is switched to +3 and the files are all copied to <svdrnm>, the LND being mapped last. After successful mapping of the LND, the workspace is deallocated and the tool session state switched to +4.

There are several possible strategies for managing the operation of the background server. It may function periodically, simply by setting a timer for inter-service intervals, or it may operate on demand. Since its primary function is to keep the set of available workspaces free to handle the beginning of new tool sessions, it is probably most effective to have it function more-or-less on demand; however, common garbage collection functions can adequately be handled with somewhat infrequent wakeup intervals. As a result of these

considerations, the strategy we are adopting is a mixture of the two approaches. Let a count of the number of unassigned workspaces be maintained in the Foreman common page. Let the background server wake up periodically (say every 15 minutes), examine this count, and, if it is not smaller than some fixed value, dismiss for another fixed interval. Let each Foreman beginning a tool session reduce this count by one. The activation level for the background server will be set on the basis of the assigned frequency of background service and the expected rates of beginning and ending tool sessions. The background server will be receptive to an interprocess signal dismissed. Let each Foreman beginning a tool session signal the background server (whose process name has been saved by itself in a block of the common page at initialization time) when it takes the next-to-the-last available workspace. This strategy seems to provide a smooth transition from a state in which the cleanup and garbage collection procedures are purely cyclical to one in which they are interrupt-driven as a function of the demand for new workspaces and the frequency of LNDSAVEs.